# Implementació d'un sistema per al control de la qualitat d'imatges JPEG

UNIVERSITAT OBERTA DE CATALUNYA
Consultor: Julià Minguillón Alfonso
Autor: Jordi Pino Lacosta
29 de juny de 2001

# Índex

# 1 Introducció

## 1.1 Convencions tipogràfiques

Les referències a estructures o rutines emprades en el projecte s'escriuen amb lletra `Courier New` de 10 punts.

Les paraules angleses i les fórmules s'escriuen en itàlica.

## 1.2 Introducció

L'estàndard JPEG és el format gràfic per a imatges fixes de to continu que s'utilitza com a mitjà de publicació en la web. La seva bona relació entre la raó de compressió obtinguda i la qualitat de la imatge comprimida, així com la seva àmplia difusió entre els usuaris de la xarxa Internet, han fet que hagi estat adoptat com a l'estàndard per a compressió d'imatges, desbancant altres formats gràfics.

La qualitat i la relació de compressió de les imatges JPEG estan determinades per les matrius de quantificació usades en el procés de compressió (cada component de la imatge pot estar quantificada amb una matriu diferent) Les matrius utilitzades s'emmagatzemen amb l'arxiu JPEG per a permetre la descodificació de la imatge amb els algoritmes adequats. Per defecte, l'estàndard JPEG proporciona una matriu de quantificació que, empíricament, s'ha demostrat que produeix imatges de qualitat mitja i/o alta i relacions de compressió bones per la majoria d'imatges i d'usuaris. Multiplicant la matriu de quantificació per un escalar s'obtenen diferents relacions de compressió i qualitats d'imatge.

Un dels problemes actuals de l'estàndard JPEG és la manca de relació entre el factor de qualitat especificat per la matriu de quantificació i la qualitat real que s'obté després del procés de compressió. A més, la percepció de l'observador és un terme completament subjectiu que habitualment és ignorat en el procés de compressió, quan realment hauria de tenir-se en compte en el moment de calcular la matriu de quantificació. El resultat és que l'única manera de saber com quedarà guardada la imatge després de tot el procés és tornant a visualitzar-la, és a dir: utilitzant el procediment d'assaig i error. Aquest procés té

un cost computacional elevat ja que la imatge s'ha de comprimir, descomprimir i després calcular l'error quadràtic mitjà[1] per a saber la qualitat d'imatge obtinguda.

És necessari, doncs, disposar d'eines que permetin experimentar modificant paràmetres de la matriu de quantificació per a aconseguir una qualitat d'imatge satisfactòria segons les percepcions de l'usuari, sense penalitzar la raó de compressió i amb un cost computacional reduït. Aquestes eines han de permetre establir el valor de qualitat desitjat i veure els resultats abans de processar la imatge definitivament.

## 1.3 Estructura de la memòria

La memòria està organitzada en 11 capítols. Contenen una explicació de les definicions i tècniques bàsiques que cal conèixer per a treballar amb imatges; l'explicació dels detalls de la implementació del projecte i dels resultats i conclusions obtingudes; la bibliografia emprada i un seguit d'annexos amb detalls de les estructures que s'utilitzen i el codi font de les rutines dissenyades.

Més concretament::

- La introducció explica les convencions tipogràfiques i explica les motivacions que fonamenten el treball.
- Al capítol "Descripció" s'expliquen els objectius i el pla de treball seguit.
- El tercer capítol, "Imatges i la seva representació", defineix una sèrie de conceptes generals i bàsics per a poder treballar amb imatges.
- El capítol "Els arxius BMP" descriu les característiques d'aquests tipus d'arxius: descripció, formats i capçaleres.
- El capítol següent fa una descripció somera del format d'imatge PNM, que inclou els formats PBM, PGM i PCM.
- El sisè capítol, "L'estàndard JPEG", introdueix al lector en la manera com es generen els arxius JPEG i en quines eines teòriques està basat.

[1] En anglès *Mean Square Error (MSE)*

- El capítol 7 explica el significat de les mesures MSE i PSNR.
- A "Model HVS" es fa una introducció a la influència del model visual humà a l'hora de comprimir imatges i de com es pot tenir en compte.
- El capítol 9, " Càlcul de la matriu Q", dóna les bases teòriques i les referències per a explicar com i perquè s'utilitza l'algoritme descrit.
- A "Implementació i estructures" es descriuen amb detall les estructures emprades; les rutines i les seves interfícies i les relacions existents entre tots aquests elements.
- Les "Conclusions" expliquen fins on s'ha arribat i quines són les previsions d'ampliació del projecte.
- Finalment, als "Annexos" es troben les definicions completes de les estructures que ha calgut utilitzar i com està relacionada la informació que guarden les estructures.

# 2 Descripció

## 2.1 Objectius

Es vol construir un sistema que permeti comprimir una imatge utilitzant el format JPEG, però incloent-hi informació que depengui de les característiques de la imatge, de la percepció de l'observador i del nivell de qualitat desitjat per l'usuari. Es tracta de satisfer aquestes premisses sense penalitzar en cap cas la raó de compressió i amb costos computacionals raonables.

L'objectiu del TFC es construir una llibreria de rutines bàsiques sobre les quals es basi una eina de software que permeti comprovar la qualitat d'un arxiu JPEG abans de guardar-ho definitivament. Aquestes rutines permetran calcular la matriu de quantificació que proporcioni la qualitat desitjada per l'usuari per a una imatge donada. L'eina complerta proporcionarà la interfície d'usuari i la visualització de la imatge abans de guardar-la definitivament. No és l'objectiu d'aquest treball implementar l'estàndard JPEG (la seva mida i complexitat ho impossibiliten), però sí diverses etapes prèvies: preprocessament, transformació i quantificació. La llibreria haurà de permetre llegir un arxiu en format BMP sense compressió, amb imatges monocromes, o de 3 components de color, i amb 1 byte d'informació per pixel. Posteriors ampliacions de la llibreria estendrien la funcionalitat a qualsevol tipus d'arxius d'imatge, així com l'augment dels paràmetres controlables per l'usuari.

Els passos bàsics que caldrà implementar són els necessaris per a obtenir una matriu de quantificació.

Aquests són:

1. Llegir la imatge original i realitzar la transformació de color (si es tracta d'una imatge a color).

2. Segmentar cada component de la imatge en blocs de 8x8 pixels i calcular la DCT de cada bloc.

3. Calcular els descriptors estadístics de cada component transformada segons un cert model probabilístic (mitjana, variància i factor de forma d'una variable aleatòria Laplaciana o Gaussiana generalitzada).

4. A partir de la qualitat desitjada, del model del sistema visual humà utilitzat i dels descriptors estadístics calculats anteriorment, calcular la matriu de quantificació utilitzant l'algorisme descrit a l'article *"JPEG standard uniform quantization error modeling with applications to sequential and progressive operation modes"* [2]

5. Un cop calculada la matriu de quantificació, cada component de la imatge transformada és quantificada i desquantificada, i després es calcula l'antitransformada de cada bloc, de manera que es simula el procés que realitza l'estàndard JPEG (excepte l'etapa de codificació ja que, en no introduir cap pèrdua i ser completament reversible, només és necessària per a generar la imatge comprimida) per a obtenir la imatge reconstruïda a partir de la imatge comprimida.

6. El procés ha de permetre diverses proves canviant algun dels paràmetres i els coeficients de la matriu de quantificació, de manera que l'usuari vegi per pantalla la imatge tal i com quedaria comprimida amb l'estàndard JPEG utilitzant la matriu de quantificació calculada.

Per a dur a terme aquest procés caldrà conèixer una sèrie de conceptes relacionats amb la representació d'imatges i dels formats dels arxius BMP i dels arxius JPEG.

[2] Veure Bibliografia

## 2.2 Pla de treball

| descripció del treball | data prevista de finalització | data real de finalització |
|---|---|---|
| Recollida d'informació i estudi del format de compressió d'imatges JPG | 25 de març de 2001 | 25 de març de 2001 |
| Recollida d'informació i estudi del format d'imatges BMP | 8 d'abril de 2001 | 8 d'abril de 2001 |
| Definició de les estructures necessàries per als mòduls | 19 d'abril de 2001 | 19 d'abril de 2001 |
| Estructura de la memòria | 20 de maig de 2001 | 20 de maig de 2001 |
| Mòdul que preprocessa la imatge segmentant-la en blocs de 8x8 i realitzant la conversió del model de color si s'escau, generant una estructura de blocs. Realitzar la conversió del model de color inversa per a la visualització | 20 de maig de 2001 | 20 de maig de 2001 |
| Mòdul que llegeix una imatge en algun format conegut (BMP) i que l'emmagatzema en memòria | 20 de maig de 2001 | 20 de maig de 2001 |
| Mòdul que rep una estructura de blocs i calcula l'etapa de transformació utilitzant la transformada discreta del cosinus. Cal poder calcular l'antitransformada | 27 de maig de 2001 | 27 de maig de 2001 |
| Mòdul que a partir d'una estructura de blocs transformats calcula els descriptors estadístics segons un model probabilístic donat | 3 de juny de 2001 | 3 de juny de 2001 |
| Mòdul que a partir d'uns descriptors estadístics i d'uns paràmetres relatius al procés (model del sistema visual humà, qualitat desitjada, i d'altres) calcula una matriu de quantificació | 6 de juny de 2001 | 6 de juny de 2001 |

| | | |
|---|---|---|
| Mòdul que a partir d'una estructura de blocs i d'una matriu de quantificació donades realitza l'etapa de quantificació (i també de desquantificació) | 10 de juny de 2001 | 10 de juny de 2001 |
| Mòdul que calcula una matriu de quantificació segons una sèrie de paràmetres i la utilitza per a simular el procés de compressió amb pèrdua que efectua l'estàndard JPEG | 20 de juny de 2001 | 23 de juny de 2001 |
| Una interfície que permeti a l'usuari llegir imatges, visualitzar-les, llegir i gravar matrius de quantificació | no s'implementa | no s'implementa |
| Ajust i proves finals | 23 de juny de 2001 | 27 de juny de 2001 |
| Memòria i documentació del TFC | 28 de juny de 2001 | 28 de juny de 2001 |

# 3 Imatges i la seva representació

## 3.1 Representació de les imatges

A la majoria de pantalles d'ordinador, la imatge que es forma a la pantalla està composada de pixels. Cada pixel ocupa una reduïda porció rectangular de la pantalla i representa només un color.

Els pixels estan organitzats en forma d'una matriu bidimensional que ocupa tota la pantalla. Degut al nombre finit de pixels que hi caben, i a la seva organització en porcions rectangulars, es produeix un escalat de les imatges que no és agradable a la vista. Per això, com més pixels pugui contenir una pantalla més petits seran els rectangles i més propera a la realitat serà la imatge.

Actualment és tendeix a treballar amb resolucions de, com a mínim, 600x800 (480. 000 pixels), sent la tendència a evolucionar cap a resolucions de 1024x768 (786. 432) i 1280x1024 (1. 310. 720) o fins i tot més elevades.

Els objectes es dibuixen a la pantalla ajustant el color dels pixels individualment. Aquest valors s'emmagatzemen en una memòria intermèdia de vídeo (*frame buffer*) que guarda, a les seves posicions, informació sobre el color de cada pixel, i a través de la qual s'actualitza la informació de la pantalla a determinats intervals de temps. Les aplicacions canvien els valors del *frame buffer* per a modificar la pantalla.

## 3.2 Imatges de vector i bitmaps

Els arxius d'imatge es poden dividir en dos classes: imatges vectorials[3] i mapes de bits[4].

Les imatges vectorials utilitzen comandes per a dibuixar formes que representaran les imatges i no estan limitades pels dispositius de sortida: un plotter dibuixarà la imatge directament i una pantalla, o una impressora làser faran una conversió de la imatge a pixels abans de representar-la. En contrapartida, les imatges vectorials no són adequades per a representar fotografies o pintures ja que necessitarien un enorme nombre

[3] En anglès es coneixen com a *vector*

[4] En anglès *bitmap* o *Bitmap GraghicsFormat* o *Raster Graphics Format*

d'instruccions i el càlcul no seria computacionalment factible. A canvi no necessiten grans quantitats de memòria per a ser emmagatzemades.

Els mapes de bits representen les imatges en forma de matrius bidimensionals on cada element representa el valor de color i la posició de la pantalla on s'ha de dibuixar un pixel determinat. Si els pixels estan suficientment a prop l'un de l'altre serà difícil per a l'ull humà apreciar la matriu i tindrà la sensació de que és una imatge contínua. L'avantatge més important d'aquest tipus d'imatge és la qualitat que es pot obtenir, especialment amb la millora dels medis d'emmagatzemament actuals, ja que el principal inconvenient és la quantitat de memòria necessària per a emmagatzemar-les. La mida en bytes d'una imatge es pot calcular com:

$$\frac{ample * alt * bitsperpíxel}{8}$$

Per exemple, una imatge amb resolució 800x600 amb 3 bytes (24 bits) per pixel necessita 1. 440. 000 bytes de memòria.

Un altre inconvenient és que els mapes de bits depenen de la mida i no es poden editar indefinidament. En canvi, les imatges vectorials permeten l'edició (escalat, modificació de perspectives, etc. ) amb molta senzillesa.

En definitiva, no es pot afirmar que un format sigui millor que l'altre. Simplement serà més adequat un o altre en funció dels requeriments de l'aplicació.

## 3.3 Models de color

Els models de color especifiquen com es representen els colors, és a dir, de quina manera guardem la informació necessària per a representar cada pixel. Depenent del tipus de dispositiu de visualització serà més adequat un o un altre.

L'objecte inicial d'aquest estudi és veure com podem controlar la qualitat de les imatges guardades en format JPEG, partint d'imatges emmagatzemades en format BMP. Les imatges JPEG acostumen a utilitzar el format de color YCbCr i les imatges BMP fan ús del format RGB.

### 3.3.1 Model RGB

El color de cada pixel és representat per la intensitat dels tres components bàsics de color: vermell (Red), verd (Green) i blau (Blue) El color blanc s'obté quan les tres intensitats són màximes i el negre per a intensitats de valor 0. Amb la combinació dels tres valors entre aquests dos límits podem obtenir qualsevol variant adequadament la intensitat de cada un; la gamma de grisos s'obté quan R = G = B.

En programació el rang de valors i, per tant la gamma de colors que es pot obtenir, està limitada pel nombre de bits que utilitzem per a cada component de color del pixel. Per a imatges fotogràfiques s'acostuma a utilitzar una precisió de 8 bits per color, tot i que en altres aplicacions també són usuals 1, 2, 4, 8, i 12 bits.

El rang de valors d'intensitat està comprès entre 0 i $2^{\text{precisió}}$-1.

### 3.3.2 Model YCbCr

Les imatges JPEG acostumen a usar el model YCbCr. Aquest model mesura, per a cada pixel, la luminància o intensitat de la llum (Y), la quantitat de component de llum blava de la imatge (Cb) i la quantitat de component de llum vermella (Cr) És un model que permet la compatibilitat amb dispositius de blanc i negre. Concretament, la component Y, per ella sola, permet la representació d'imatges en escala de grisos.

### 3.3.3 Conversió de models

La relació típica de conversió entre els models és:

RGB a YCbCr :

$$Y = 0.299\,R + 0.587\,G + 0.114\,B$$
$$Cb = -0.1687\,R - 0.3313\,G + 0.5\,B + 2^{\text{precisió}/2}$$
$$Cr = 0.5\,R - 0.4187\,G - 0.0813\,B + 2^{\text{precisió}/2}$$

YCbCr a RGB:

$$R = Y + 1.402\,(Cr - 2^{\text{precisió}/2})$$
$$G = Y - 0.34414\,(Cb - 2^{\text{precisió}/2}) - 0.71414\,(Cr - 2^{\text{precisió}/2})$$
$$B = Y + 1.772\,(Cb - 2^{\text{precisió}/2})$$

Però aquest valors els podem modificar per aconseguir una imatge més adaptada a les característiques particulars de visió de cada persona. Les llibreries permetran

emmagatzemar diferents valors en una matriu, de manera que es podran modificar a voluntat de l'usuari. Per defecte s'utilitzen els descrits més amunt.

## 3.4 Color vertader i paleta de colors

Normalment s'utilitzen dos mètodes per a representar els colors d'un pixel. el més senzill és guardar el valor del color amb la resta de dades de la imatge. En el cas d'imatges monocromes un byte pot donar informació sobre el color d'un pixel. Si usem 24 bits (3 bytes) obtenim imatges de color vertader (true color), 224= 16777216 colors diferents, que és el límit de colors que pot distingir l'ull humà.

Els ordinadors més antics tenien problemes de capacitat per a processar aquestes elevades resolucions de color. Per a solucionar-ho s'utilitzen paletes de colors, que no és rés més que una matriu unidimensional de estructures de 3 bytes que especifiquen un color. Llavors, es treballa només amb un subconjunt de colors i el valor de cada pixel és un índex que referencia una posició de la matriu. La mida més comú per a una paleta és de 256 entrades i el valor de cada pixel és el que hi ha especificat en una entrada concreta de la taula. D'altres mides poden ser 1, 2, o 4 bits per pixel, aleshores cada entrada de la taula guarda la informació de color de 8, 4 o 2 pixels. Els arxius BMP poden utilitzar qualsevol d'aquests formats.

## 3.5 Compressió

### 3.5.1 Formats de compressió

La majoria d'arxius d'imatge utilitzen tècniques de compressió per a disminuir la grandària dels arxius. Algunes d'aquestes són:

- *Run Lenght Encoding* (RLE): pixels consecutius amb el mateix valor es representen amb una parella de valors, el primer component expressa el valor que es repeteix i el segon diu quantes vegades es repeteix.

- Codificació LZ: es manté un diccionari que conté els valors dels pixels de la imatge. Els bytes que representen la imatge comprimida fan referència a entrades del diccionari.

- Codificació Huffman: en lloc d'utilitzar un nombre fix de bits per a representar els valors dels components, s'usen codis de longitud variable, de manera que com més es repeteix un valor més curt és el codi que el representa.

- Transformada Discreta del Cosinus (DCT): es representen blocs de pixels usant funcions cosinuidals de diferents freqüències. Les altes freqüències, que generalment són les que aporten menys informació a la imatge, són descartades.

Cada tipus de compressió és més o menys efectiu depenent del tipus de dada.

### 3.5.2 Compressió sense pèrdues (*lossless*) i amb pèrdues (*lossy*)

La majoria de formats d'arxiu utilitzen la compressió d'imatges sense pèrdues (*lossless*), la qual cosa significa que un cop descomprimida es recupera la imatge exactament igual a l'original bit a bit.

Alguns mètodes de compressió pateixen pèrdues en el procés de compressió (*lossy*) sent impossible la recuperació exacte de la imatge original. La reconstrucció és gairebé igual, però no exactament, a canvi podem aconseguir relacions de compressió més elevades. Els formats que utilitzen tècniques de compressió amb pèrdues juguen amb la propietat de l'ull humà que no permet apreciar massa bé les petites variacions de colors similars.

## 3.6 Bytes i ordre dels bits

Els arxius d'imatge contenen enters emmagatzemats en format binari. Quan s'utilitzen enters que ocupen un sol byte no hi ha problemes de compatibilitat entre diferents tipus de processadors, però aquest no és el cas del enters que es representen amb més d'1 byte.

La família de processadors Motorola 680x0 i Sun Sparc guarden la informació en "ordre de xarxa" (dita així perquè és l'ordre que usen els protocols d'Internet) o format *big endian* que consisteix en guardar el byte més significatiu en primer lloc. En canvi, els processador de la família Intel 80x86 utilitzen el format *little endian*, que guarden el byte menys significatiu en primer lloc.

Aquests aspectes els haurem de tenir en compte si, en un futur, es decidís fer aquestes rutines portables a sistemes diferents del triat per aquest projecte: processadors de la família Intel 80x86.

# 4 Els arxius BMP

## 4.1 Què són els arxius BMP

El format BMP és un dels més senzills i és el format nadiu d'imatges en el sistemes operatius de Microsoft Windows. Pot treballar amb imatges d'1, 4, 8, 24 i 32 bits per pixel, tot i que són estranyes les aplicacions que treballen amb 16 i 32 bits. BMP també suporta compressió *Run Lenght Encoding* (RLE) per 4 i 8 bits per pixel, encara que tampoc és habitual trobar imatges que en facin ús.

## 4.2 Estructura dels arxius BMP

L'estructura dels arxius BMP és força simple, tal com mostra la figura 1.


**Figura 1. Estructura d'un arxiu BMP**

`BITMAPFILEHEADER` conté informació sobre el tipus, mida i lloc on comencem les dades de la imatge.

`BITMAPINFOHEADER` defineix característiques de la imatge com ara la mida, el nombre de colors...

RGBQUAD guarda la taula de colors emprada.

BYTE emmagatzema els valors que representen la intensitat de color dels pixels de la imatge.

Una descripció més detallada de tots els camps es pot trobar a l'annex 2.

## 4.3 Conclusió

El format BMP triat és un bon format per a poder fer experimentacions, ja que no ens introdueix complicacions addicionals a les inherents al projecte.

En una primera fase, aquest projecte està preparat per a tractar amb imatges BMP de 8 o 24 bits per pixel, ja que les imatges de 1 i 4 bits per pixel no permeten ésser comprimides amb el format JPEG i les de 16 i 32 bits són poc utilitzades. L'elecció de només imatges monocromes (8 bits) o de color (24 bits) no impedeixen el desenvolupament futur de les rutines. Tant sols caldrà afegir lleugers augments en les funcionalitats.

# 5 Els arxius PNM (PBM, PGM i PPM)

## 5.1 Què són els arxius PNM (PBM, PGM i PPM)

Els formats PBM, PGM i PPM s'utilitzen per a emmagatzemar imatges en blanc i negre, d'escala de grisos i de color, respectivament.

S'identifiquen pel valor `MagicValue` de la seva capçalera. Per a cada un dels formats hi ha una versió del valor en binari i un altre en ASCII. Cada pixel es representa amb 3 bytes que proporcionen informació sobre els seus valors RGB.

Les imatges PBM només poden ser llegides si l'amplada d'imatge és múltiple de 8

## 5.2 Estructura dels arxius PNM (PBM, PGM i PPM)

L'estructura dels arxiusPNM encara és més simple que les del BMP, com es veu a la figura 2.

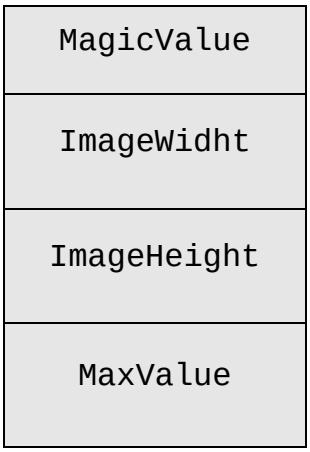



**Figura 2. Estructura d'un arxiu PNM**

Els valors que pot prendre `MagicValue` són:

| format | ASCII | bits (RAW) |
|---|---|---|
| **PBM** | P1 | P4 |
| **PGM** | P2 | P5 |
| **PCM** | P3 | P6 |

El camp `MaxValue` no s'utilitza en el cas d'imatges PBM.

## 5.3 Conclusió

Aquest tipus d'arxiu ofereix una gran senzillesa i portabilitat entre sistemes, per la qual cosa es creu convenient iniciar les futures ampliacions del projecte donant suport a aquest format.

# 6 L'estàndard JPEG

## 6.1 Què són els arxius JPEG

JPEG és l'acrònim "Join Photographic Experts Group", organització que ha desenvolupat aquest estàndard per a la compressió d'imatges. És un format propietari, però han aparegut d'altres organitzacions que han desenvolupat aplicacions que no utilitzen els mateixos algoritmes ni patents, però que són completament compatibles amb les especificacions estàndards, per exemple "Independent JPEG Group".

JPEG està dissenyat per treballar amb imatges fotogràfiques o artístiques, no donant tant bons resultats amb imatges de línies, de dibuixos senzills o bé de text.

Els arxius JPEG són els més utilitzats per a emmagatzemar imatges fotogràfiques degut a les seves característiques de relació entre la qualitat d'imatge i la compressió de l'arxiu.

**Figura 3**

**Figura 4**

Les imatges anteriors s'han obtingut a partir d'una imatge BMP de resolució 447x 667 pixels i 24 bits de color. La mida original és de 909.942 bytes. La imatge de la figura 3 s'ha obtingut amb un factor de compressió de 95 i ocupa 167.253 bytes. Per la imatge de la figura 4 el factor de compressió és de 25 i la mida 29.169 bytes. En els dos casos la resolució continua sent de 447x 667 pixels i 24 bits de color.

## 6.2 Introducció a l'estàndard JPEG

L'estàndard JPEG per a imatges fixes de to continu (és tal i com està definit i tal com s'anomena) es fonamenta en les característiques d'aquest tipus d'imatges i del sistema visual humà. Concretament:

- Les imatges de to continu, (o naturals, en contra de les imatges sintètiques com ara plànols, text, etcètera) presenten degradacions suaus: hi ha una alta correlació entre el valor d'un pixel de la imatge i tots els que l'envolten, excepte quan hi ha un canvi brusc degut a dos objectes diferents, per exemple.

- La sensibilitat de l'ull humà a certes freqüències i contrastos no és uniforme, sinó que és variable.

L'estàndard JPEG combina aquests dos fets de manera que és possible reduir la quantitat d'informació necessària per a representar el senyal (la imatge) seleccionant tan sols aquella informació visual que és veritablement necessària sota un cert criteri de qualitat.

La qualitat de la imatge comprimida queda determinada per la quantitat de senyal eliminada en el procés de compressió. L'estàndard JPEG utilitza un procés de compressió amb pèrdua (*lossy*), de manera que la imatge comprimida no és idèntica a l'original sinó només una reconstrucció amb un cert grau de fidelitat. En general, un procés de compressió amb pèrdua està composat per les següents etapes:

- Preprocessament: la imatge és segmentada en blocs (8x8 en el cas de l'estàndard JPEG) i convertida a un model de color adient (YCbCr).

- Transformació: cada bloc és transformat de manera que l'energia de cada bloc (el senyal) es concentra en pocs coeficients (l'estàndard JPEG utilitza la transformada discreta del cosinus, o DCT).

- Quantificació (o quantificació): els coeficients són discretitzats de manera que passen a tenir una representació finita (l'estàndard JPEG utilitza una quantificació escalar uniforme).

- Codificació: els valors generats en l'etapa anterior són agrupats de manera que s'intenta reduir el nombre de bits necessaris per a representar-los (l'estàndard JPEG permet tan la codificació aritmètica com la codificació Huffman).

A diferència del que succeeix en un procés de compressió sense pèrdua (*lossless*), tant la qualitat de la imatge com la raó de compressió (la raó entre la mida de la imatge original i la imatge comprimida) són variables, i queden determinades per les operacions realitzades en cadascuna de les etapes esmentades anteriorment. No obstant això, l'única etapa que admet ser fàcilment parametritzada és la de quantificació. En el cas concret de l'estàndard JPEG, cada coeficient dels 64 que genera la transformada discreta del cosinus d'un bloc de 8x8 és discretitzat mitjançant una quantificació uniforme, de manera independent de la resta de coeficients (d'aquí l'adjectiu escalar). Aquesta quantificació consisteix en dividir per un coeficient enter i quedar-se amb la part entera del resultat obtingut fent un arrodoniment.

Suposem per exemple que es quantifica utilitzant un valor Q=5. Aleshores, qualsevol valor entre –2. 5 i +2. 5 passa a valer 0, qualsevol valor entre +2. 5 i +7. 5 passa a valer +1, qualsevol valor entre –7. 5 i –2. 5 passa a valer –1, etcètera. La reconstrucció (o desquantificació) consisteix en multiplicar de nou pel valor Q, de manera que el valor +1 es reconstrueix per 5, per exemple. S'observa que tot un interval de valors dels coeficients originals com ara [2. 5, 7. 5) es converteix en un únic valor enter (+1, o 5 un cop desquantificat). Això provoca la pèrdua que de fet, és la que permet assolir una raó de compressió elevada.

Els sistemes de compressió sense pèrdua d'imatges fixes de to continu només permeten assolir raons de compressió properes de fins 5:1 (la imatge comprimida ocupa 5 cops menys que la imatge original). En el cas concret de l'estàndard JPEG, és fàcil assolir raons de compressió al voltant de 15:1 sense una gran degradació de la qualitat de la imatge comprimida.com és possible? Doncs aprofitant que l'energia dels 64 pixels de cada bloc queda molt concentrada en uns pocs dels 64 coeficients, i aquests són tractats de manera que aquelles freqüències on la resposta del sistema visual humà és pitjor són més quantificades. La idea bàsica és no utilitzar cap bit per a representar informació present en la imatge original que l'ull humà no pot percebre.

Queda clar, per tant, que el procés de quantificació és crític, ja que determina tant la raó de compressió que s'aconsegueix com la qualitat de la imatge comprimida. En el cas de l'estàndard JPEG això s'aconsegueix mitjançant una matriu de quantificació predefinida per a cada component de la imatge i que conté els 64 valors Q utilitzats (un per cada coeficient). Avui dia encara no hi ha mecanismes que de manera senzilla permetin calcular matrius de quantificació adaptades a cada imatge (i de fet a cada observador). Manquen, doncs, eines que permetin comprimir una imatge donada amb una qualitat desitjada a partir de la generació de matrius de quantificació personalitzades.

Gràficament el procés que s'ha de seguir per a guardar una imatge en format JPEG és el que mostra la figura 5.

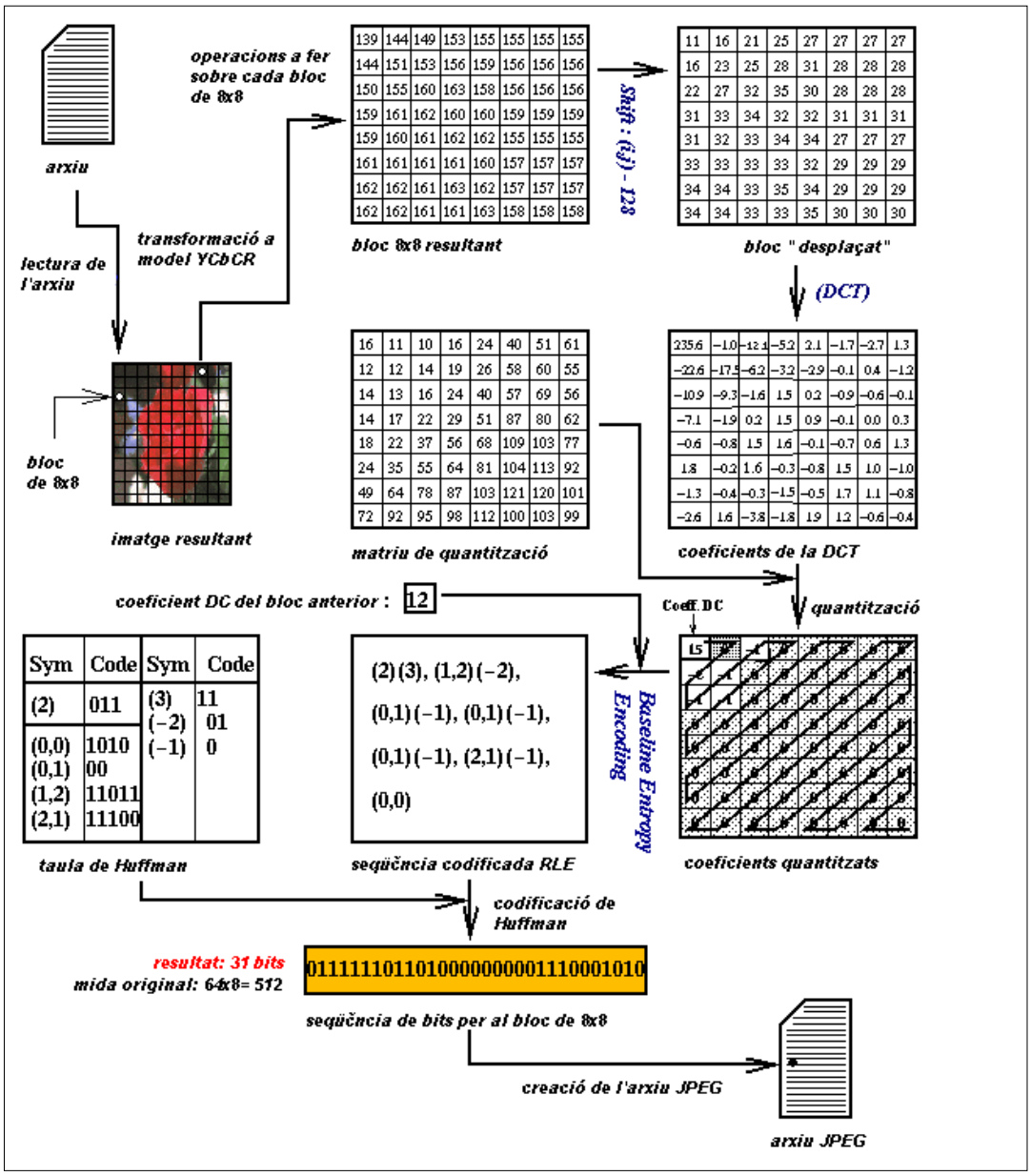



**Figura 5. El procés de compressió dels arxius JPEG**

## 6.3 Modes de compressió JPEG

L'estàndard JPEG defineix quatre modes de compressió d'arxius: compressió *hierarchical*, progressiva, seqüencial i sense pèrdues (*lossless*)

- La compressió seqüencial és la més utilitzada i codifica les imatges d'una sola passada i des del principi fins al final de l'arxiu.

- En el mode progressiu la imatge s'escaneja en successives passades, entre 2 i 896, refinant les dades a cada escanejada. A la descodificació es produeix l'efecte invers i es pot veure la millora de qualitat de la imatge a cada escaneig.

- El mode *hierarchical* és un mode super-progressiu on la imatge es descomposta en subimatges anomenades trames. La primera trama crea una versió de molt baixa qualitat i la resta refinen la imatge incrementant la resolució.

- El mode *lossless* simplement conserva la imatge original exacte i ha quedat obsolet.

## 6.4 La Transformada Discreta del Cosinus (DCT)

La Transformada Discreta del Cosinus dóna informació sobre la freqüència dels valors emmagatzemats als blocs de 8x8. Transforma un conjunt de valors d'entrada en un altre conjunt de valors de funcions cosinuidals amb freqüències creixents. Això permetrà eliminar aquelles components de freqüència per a les quals l'ull humà no té tanta sensibilitat, concretament les altes, sense modificar les components de baixa freqüència. La transformada inversa DCT no produeix una pèrdua significant d'informació, tret dels errors per arrodoniments efectuats durant la transformació directa.

Abans d'aplicar la DCT es fa una operació de desplaçament a l'esquerra de 128:

$$I(i, j)=I(i, j) -128$$

D'aquesta forma el rang de valors està comprés entre –128 i +128 en lloc d'entre 0 i 255

La DCT s'aplicarà sobre aquesta matriu, que està subdividida en blocs de 8x8. En el cas que hi hagi files o columnes que no siguin múltiples de 8 s'hauran de completar amb zeros, ja que la DCT s'ha d'aplicar a blocs complerts de 8x8.

El primer coeficient obtingut amb la DCT es coneix amb el nom de coeficient DC, la resta són els coeficients AC. El coeficient DC mesura el valor mitjà dels 64 mostres de la imatge i els coeficients AC aporten informació sobre la freqüència. Ambdós coeficients es tractaran de diferent manera en la codificació JPEG.

## 6.5 Quantificació

L'objecte de la quantificació és aconseguir el màxim de compressió representant els components de la DCT amb el mínim de resolució possible tal que permeti obtenir la qualitat d'imatge desitjada.

Els valors de quantificació idealment han de ser triats de forma que s'adeqüin al llindar de percepció (el límit d'apreciació de la diferència entre imatges) ja que no té sentit fer diferenciacions de valors que després no puguin ésser apreciats. Aquest llindar depèn de les característiques de la imatge original, de les característiques dels dispositius de visualització i de la distància de visió.

Qualsevol bloc 8x8 al qual se li ha aplicat la DCT es quantificarà segons l'expressió:

$$F^Q(u, v)=round(F(u, v)/Q(u, v))$$

on *Q(u, v)* és la matriu de quantificació i F(u, v) és la matriu dels coeficients de la DCT.

Escalant els valors de la matriu de quantificació es pot variar la quantitat d'informació eliminada de la imatge. La variació pot estar entre 1 (imatges de poca qualitat a causa de l'eliminació de molta informació) i 100 (imatges de gran qualitat amb poca pèrdua d'informació) Tot i això els valors es poden reduir al rang entre 25 i 95, ja que un factor de 25 ja dóna imatges de molt baixa qualitat i valors superiors a 95 produeixen imatges grans sense una millora apreciable de qualitat. Proporciona i actuarà de la següent forma sobre la matriu de quantificació. L'escalat s'aplica a cada un dels coeficients de la matriu de quantificació *Q(u, v)*:

$$Q^s(u, v)= (Q(u, v)* escalat+ 50) 100)$$

$Q^s(u, v)$ és la nova matriu de quantificació 8x8 escalada.

A la fase de descodificació, la desquantificació els coeficients de la DCT es calculen segons:

$$F^Q(u, v)=F^Q(u, v)\ Q(u, v)$$

Aquest és el mètode que utilitzant la major part de les llibreries JPEG de l'IJG (Independent JPEG Group), que es poden trobar a Internet a l'adreça: http://www.ijg.org

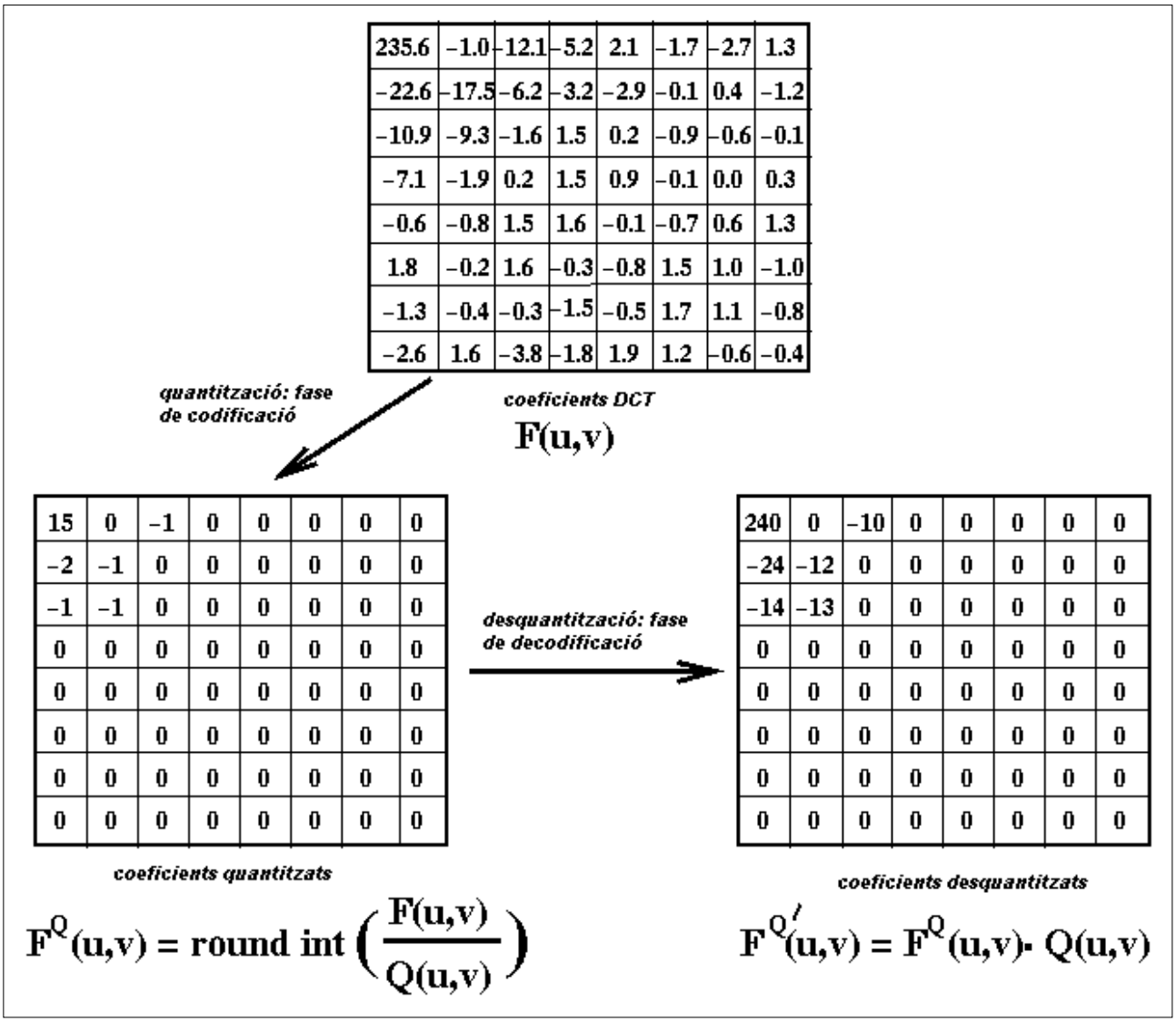



**Figura 6. Quantificació dels coeficients DCT**

Una part important d'aquest projecte es centrarà en el càlcul dels coeficients de les matrius de quantificació, tenint en compte la qualitat desitjada i el sistema visual humà. Per defecte s'utilitzaran matrius de quantificació que s'han elaborat empíricament i donen uns resultats acceptables en la majoria dels casos, tot i que no tenen en compte les peculiaritats i l'entorn d'un usuari en particular.

## 6.6 Ordenació en Ziga-Zaga

Després de la quantificació apareixen molts coeficients AC amb valor igual a 0. Ens interessa tenir seqüències de zeros tant llargues com sigui possible per a aconseguir la

màxima compressió en els processos posteriors. Per això s'ordenen seguint el recorregut en ziga-zaga (ZZ)

El coeficient DC és tractat de manera independent dels altres 63 coeficients.

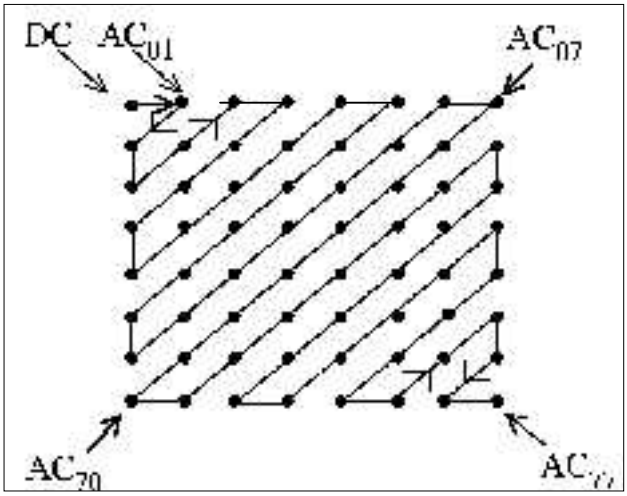



**Figura 7. El recorregut de la seqüència ziga-zaga (ZZ)**

Aquesta ordenació situa els coeficients que representen les baixes freqüències, i que tenen major probabilitat de tenir valors diferents de 0, abans dels coeficients d'alta freqüència. Aconseguim doncs tenir els valors propers a 0 junts, la qual cosa facilitarà l'eliminació de valors en el moment de la codificació Huffman.

## 6.7 Codificació Huffman pels arxius JPEG

Per a codificar els coeficients després de la quantificació s'usa primer una codificació del tipus Run Length Encoding, després de la qual es pot aplicar la codificació de Huffman. En aquest procés no es produeix pèrdua d'informació.

# 7 MSE i PSNR

MSE és una mesura de la diferència existent entre dues variables aleatòries Laplacianes. Ens serveix per a mesurar l'error quadràtic mig existent entre dos conjunts de valors aleatoris. Anomenem *X* a una variable aleatòria Laplaciana amb mitjana zero; $X_1$ una versió quantificada amb un pas Q de *X*, i *X'* la reconstrucció de *X* a partir de $X_1$. L'MSE degut a la quantificació es pot mesurar amb la fórmula:

$$MSE = E\left[\left(X - X'\right)^2\right] = \int_{-\infty}^{+\infty} (x - x')^2 f_X(x)dx \text{ , i}$$

$$X_1 = \left\lfloor \frac{X}{Q} + \frac{1}{2} \right\rfloor \text{, i}$$

$$X' = QX$$

Normalment és més fàcil per a l'usuari especificar un valor de pic de la relació senyal-soroll[5] (PSNR), ja que és una mida d'ús més freqüent a la literatura. Els dos valors estan relacionats per l'expressió:

$$PSNR = 10\log_{10}\frac{255^2}{MSE}$$

[5] PSNR: *Peak Signal Noise Ratio*, valor de pic de la relació senyal-soroll

# 8 Model HVS (*Human Visual Response*)

Si volem aconseguir matrius de quantificació ajustades al màxim cal poder incloure en el seu càlcul el model de resposta del sistema visual humà (HVS) Aquest model es basa en la sensibilitat de l'ull al nivell d'il·luminació de fons i a les freqüències del espectre visible.

La manera d'introduir aquesta correcció serà en el moment del càlcul dels coeficients de quantificació en funció del PSNR a través de les fórmules que mesuren l'error de quantificació.[6]

El model HVS es representa a través de la funció $\Phi(u,v)$ i està directament relacionat amb la sensibilitat de l'ull humà. Per tant, és lògic pensar que l'error de quantificació serà més visible per a aquelles freqüències per a les quals l'ull té major sensibilitat, això és, on el model HVS produeix els valors més alts. $\Phi(u,v)$ és inversament proporcional a l'MSE:

$$MSE_{u,v} = \frac{MSE}{\Phi(u,v)}$$

La funció $f$ proporciona un mapatge de cada coeficient $u,v$ a una freqüència, segons l'equació:

$$\Phi(u,v) = \left[0.9 + 0.18 f(u,v)\right] e^{-0.12 f(u,v)}, \text{ on}$$

$$f(u,v) = f_{màx} \frac{ZZ^{-1}(u,v)}{N^2 - 1}$$

No modelarem la resposta real del HVS, sinó que distribuirem una certa quantitat de manera aproximada i sense grans diferències entre el repartiment entre tots els coeficients. En concret utilitzarem $f_{màx} = 20$.

[6] Fórmula núm.17 de *"JPEG standard uniform quantization error modeling with applications to sequential and progressive operation modes"*

# 9 Càlcul de la matriu Q

La justificació de les fórmules que apareixen a continuació, dels valors que es prenen per defecte i de l'algoritme es pot consultar a l'article de Julià Minguillón i Jaume Pujol, *"JPEG standard uniform quantization error modeling with applications to sequential and progressive operation modes"*

Interessa que els coeficients de quantificació *u,v* s'obtinguin a partir d'un determinat PSNR, que està relacionat amb el MSE mitjançant la fórmula:

$$MSE = \frac{255^2}{10^{\frac{PSNR}{10}}} \quad (1)$$

d'on obtenim l'expressió de PSNR:

$$PSNR = 10 \log_{10} \frac{255^2}{MSE} \quad (2)$$

També es vol distribuir el MSE entre tots els coeficients. Per això caldrà conèixer el valor del MSE màxim. El MSE de cada coeficient ha d'estar comprés entre els valors:

$$MSE_{u,v} \geq MSE(1, \sigma_{u,v}) \quad (3)$$

$$MSE_{u,v} \geq mín\{S_{u,v,}MSE(255, \sigma_{u,v}) \quad (4)$$

on $S_{u,v}$ representa el valor de l'estratègia de quantificació. Com a valor per defecte es prendrà:

$$S_{u,v} = \sigma^2_{u,v} \quad (5)$$

En primer lloc s'han de calcular, doncs, els descriptors estadístics per a cada bloc de 8x8 de cada component.

A continuació s'obté l'MSE màxim, a partir de les equacions 3 i 4 i després es calcula l'MSE objectiu segons la fórmula 1. Com que volem distribuir l'HVS entre tots els coeficients i s'han de complir les equacions 3 i 4. Direm que un coeficient és saturat quan

no satisfaci l'equació 4. A causa de l'elecció feta a 5, un coeficient saturat tindrà un coeficient de quantificació igual a zero.

Altres equacions que seran útils ens donen una relació entre MSE i els coeficients de quantificació.

$$Q(MSE,\sigma) = \sigma\sqrt{2}F^{-1}(1-\frac{MSE}{\sigma^2}) \text{ , on} \qquad (6)$$

$$F(t) = \frac{t}{\sinh(t)} \text{, i } F^{-1}(x) \text{ és el valor tal que } F(t) = x$$

També s'utilitzarà l'equació:

$$Q(MSE_{0,0}) = 1, \qquad \text{si } MSE_{0,0} \leq 4.49257 \qquad (7)$$

$$Q(MSE_{0,0}) = \frac{-a_1 + \sqrt{a_1^2 - 4a_2(a_0 - MSE_{0,0})}}{2a_3} \text{ , a la resta de casos}$$

L'algoritme que s'usarà per al càlcul dels coeficients és el següent[7]:

[7] Extret de *"JPEG standard uniform quantization error modeling with applications to sequential and progressive operation modes"*

```
pns= 64; MSEdist= 64* MSEobj; s= 64; saturat(u,v)= FALSE   ∀u,v
repetir
  i= pns; flag= FALS
    repetir
     obtenir u,v a partir de ZZ(i)     /* ZZ és la seqüència zig-
zag */
     si ¬saturat(u,v)
       MSEu,v= (MSEdist/s)/Φ[f(u,v)]
       si MSEu,v > MSEu,v
          MSEu,v= MSEmàxu,v; saturat(u,v)= CERT; flag= CERT
          MSEdist= MSEdist- MSEu,v; s= s- (1/Φ[f(u,v))
          si i= pns
            pns= pns- 1
          fisi
       fisi
     fisi
     si ¬flag
       i= i- 1
     fisi
    fins flag ∨ (i<1)
fins i<1
calcular Q(u,v)    /* segons les fórmules 6 i 7 */

.
```

# 10 Implementació

Els mòduls que és necessari implementar per a la realització del projecte són els següents:

**M1** Un mòdul que llegeix una imatge en algun format conegut (o fins i tot en format RAW) i que l'emmagatzema en memòria.

**M2** Un mòdul que donada una imatge en memòria la visualitza.

**M3** Un mòdul que preprocessa la imatge segmentant-la en blocs de 8x8 i realitzant la conversió del model de color si s'escau, generant una estructura de blocs. Evidentment cal poder realitzar la conversió del model de color inversa per a la visualització.

**M4** Un mòdul que rep una estructura de blocs i calcula l'etapa de transformació utilitzant la transformada discreta del cosinus. Cal poder calcular l'antitransformada.

**M5** Un mòdul que a partir d'una estructura de blocs transformats calcula els descriptors estadístics segons un model probabilístic donat.

**M6** Un mòdul que a partir d'uns descriptors estadístics i d'uns paràmetres relatius al procés (model del sistema visual humà, qualitat desitjada, i d'altres) calcula una matriu de quantificació.

**M7** Un mòdul que a partir d'una estructura de blocs i d'una matriu de quantificació donades realitza l'etapa de quantificació (i també de desquantificació).

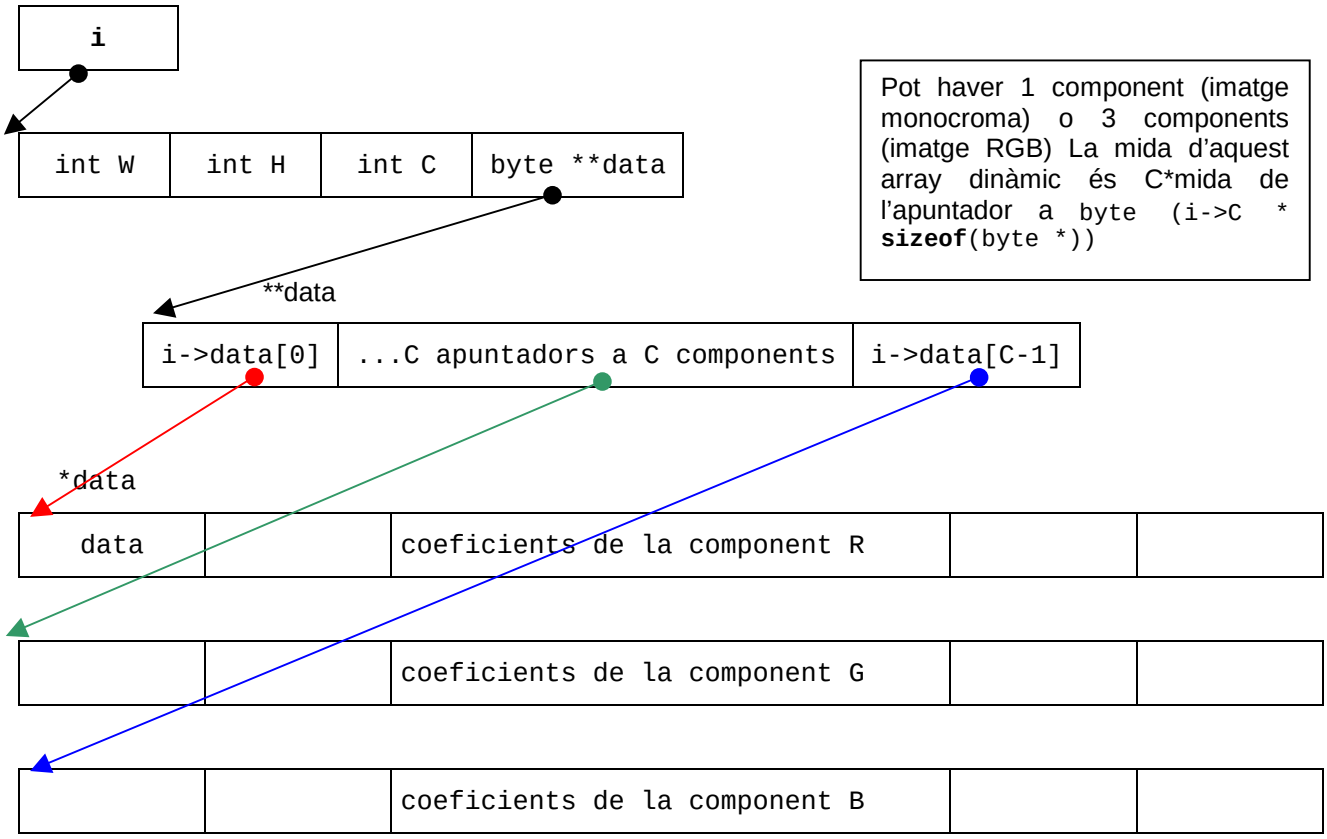
**M8** Una interfície que permeti a l'usuari llegir imatges i visualitzar-les, llegir i gravar matrius de quantificació, i realitzar els passos necessaris per a calcular una matriu de quantificació segons una sèrie de paràmetres i utilitzar-la per a simular el procés de compressió amb pèrdua que efectua l'estàndard JPEG.

## 10.1 Descripció de les estructures

A continuació es descriuen les principals estructures utilitzades per a la implementació del projecte.

`ImatgeRAW`: conté les mides de la imatge (H files, W columnes), el nombre C de components (p.e. imatge monocroma C=1, imatge RGB C=3) i un array dinàmic de C components on cada component és un apuntador a una seqüència de FC bytes.

```
typedef unsigned char byte;
typedef struct imatgeRAW {
     int W, H, C;    /* W(ample), H(alçada), C(components) */
     byte **data;    /* array d'apuntadors a les dades de la imatge
*/
} imatgeRAW;
imatgeRAW *i;
```




mida de l'array dinàmic on es guarden els valors de la imatge:

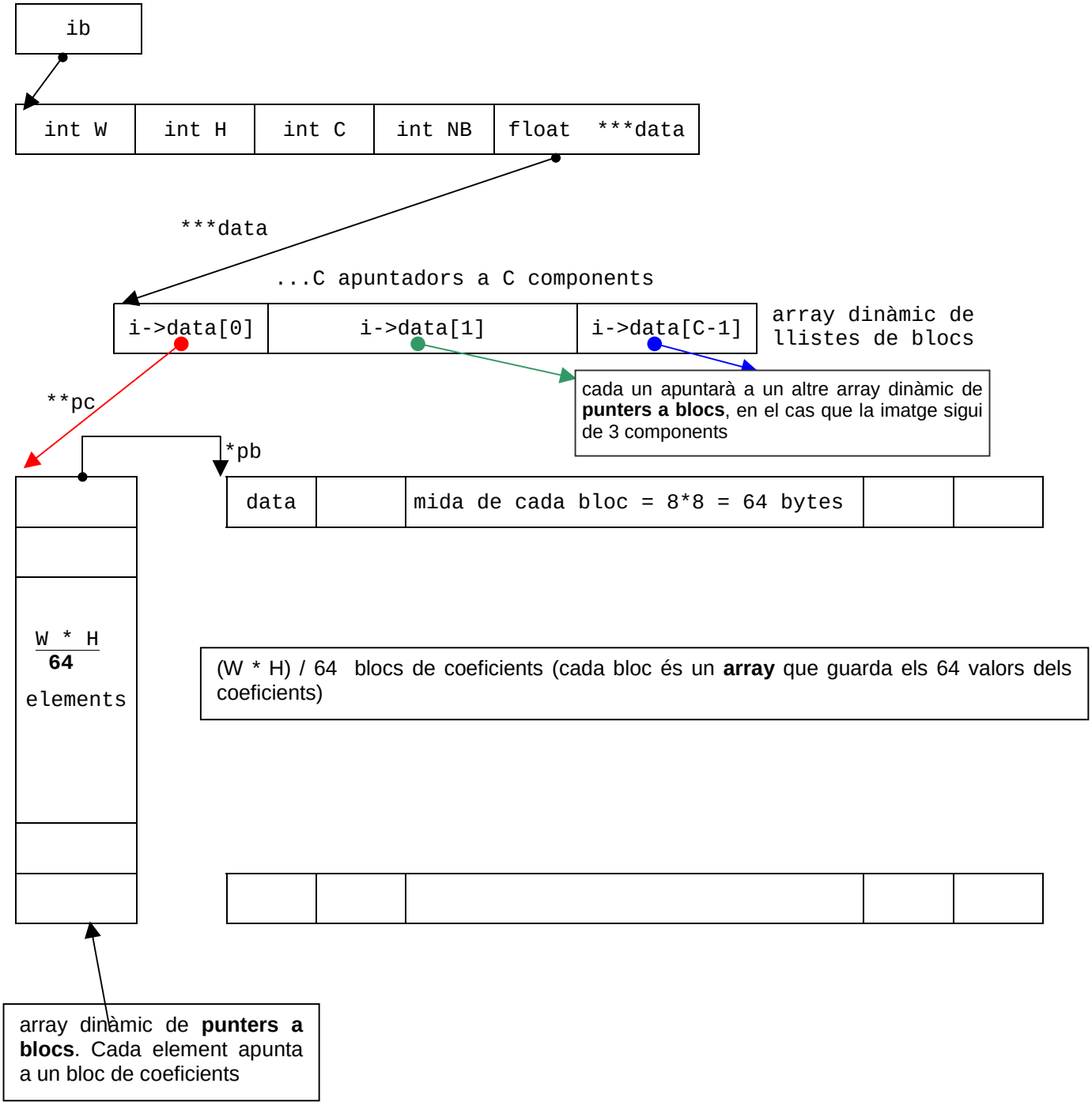
`W * H * mida byte = W * H * sizeof (byte) = W * H * sizeof (char)`

### 10.1.1 Estructura imatgeBLOCS

`ImatgeBLOCS`: conté la mateixa informació que `imatgeRAW` (H, W, C) amb algunes variacions. Disposa d'un camp esDCT que informa de si els coeficients ja han estat transformats amb la DCT o no. En lloc d'un array dinàmic de components té un array

dinàmic de llistes de blocs, on cada llista de blocs (una per cada component) és un array dinàmic de punters a bloc, i cada bloc és un array per emmagatzemar els 64 coeficients. Els coeficients ja no són pixels originals sinó que s'ha fet la conversió de color.

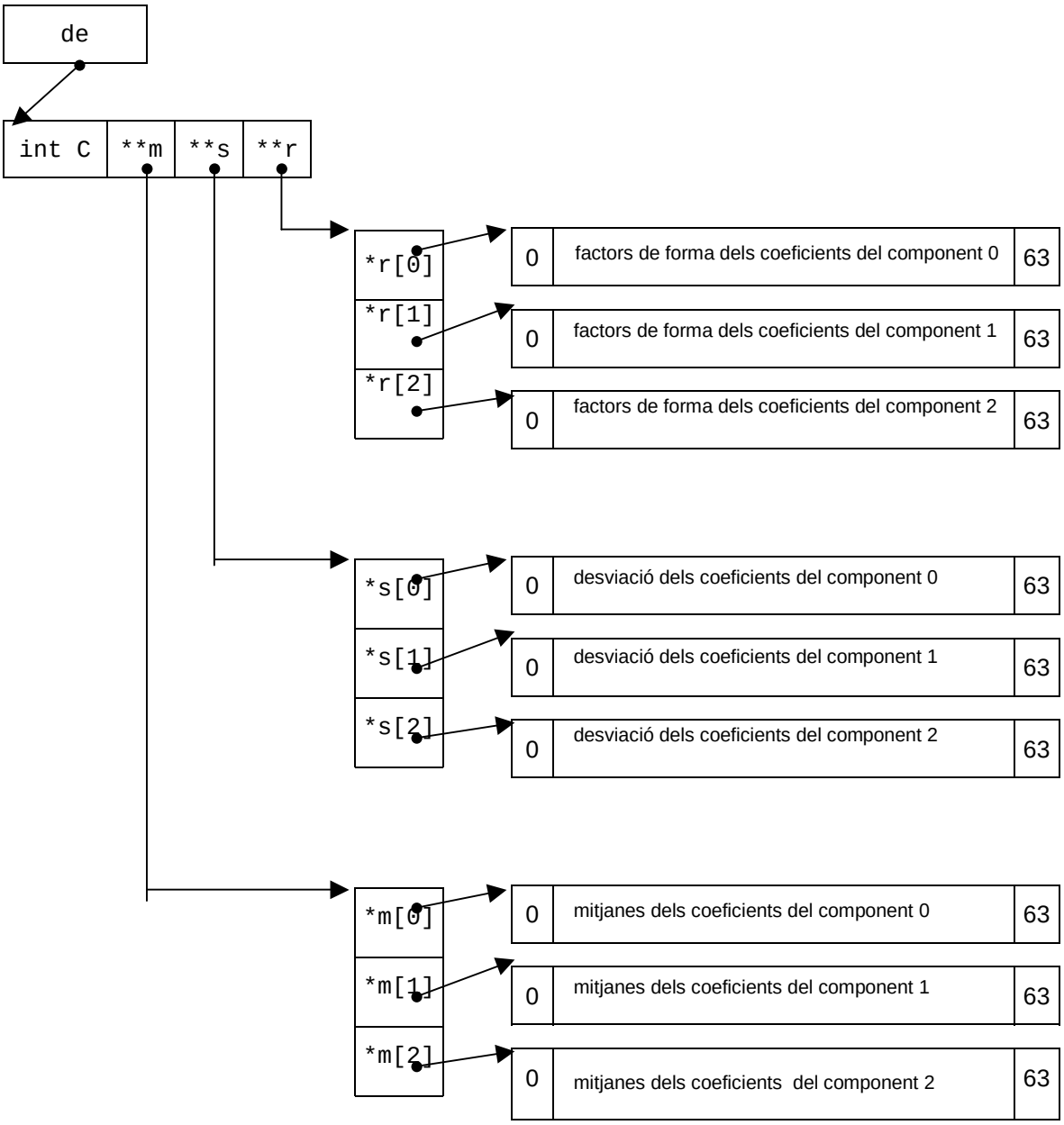
```
typedef  struct imatgeBLOCS {
     int W, H, C, NB;     /* W(ample), H(alçada), C(components) */
     float ***data; /* apuntadors als arrays de punters a blocs */
     byte esDCT;    /* cert si la imatge ha estat transformada */
} imatgeBLOCS;

imatgeBLOCS *ib;
float **pc, *pb; /* punter a components, punter a bloc */
```




### 10.1.2 Estructura descEstadistic:

`descEstadistic`: conté la informació dels descriptors estadístics d'una estructura `imatgeBLOCS`. Té tres apuntadors a arrays dinàmics que, al seu torn, contenen apuntadors a arrays on es guarden els valors estadístics associats a cada coeficient de l'estructura `imatgeBLOCS`

```
typedef struct descEstadistic{
  int C;      /* nombre de components */
  float **m;  /* apuntadors als arrays d'apuntadors a l'array de mitjanes */
  float **s;  /* apuntadors als arrays d'apuntadors a l'array de desviacions */
  float **r;  /* apuntadors als arrays d'apuntadors a l'array de factors de forma */
} descEstadistic;
```




La descripció gràfica de les relacions entre les informacions guardades a les estructures BMP, imatgeRAW i imatgeBLOCS es troba a l'annex 1.

## 10.2 Descripció de les rutines

Les rutines i les funcions implementades ofereixen les funcionalitats que es descriuen a continuació:

- *blocsquantifica.c*: rep un punter a `imatgeBLOCS` i a un array de matrius de quantificació, una per a cada component. Si no se li passa cap punter a l'array de matrius utilitza per defecte l'array de matrius definit a *estructures.h* Retorna un punter a una nova estructura `imatgeBLOCS` que conté el valor dels coeficients ja quantificats

- *conversiocolors.c*: rutines per a fer la conversió de color de RGB a YCbCR, i viceversa, entre estructures `imatgeRAW` i `imatgeBLOCS`. Inclou les funcions *obtenirYCbCR* i *obtenirRGB*, pel canvi de model de color.

- *obtenirYCbCR*: rep un apuntador a una estructura `imatgeRAW` i la posició del coeficient que volem transformar (coeficients de `imatgeRAW` en format RGB) i, utilitzant els valors de la matriu `matriuYCbCr`, obté la transformació als valors originals YCbCr per a guardar-los en una `imatgeBLOCS` . Retorna un apuntador a una matriu `YCbCr` on es guarden els 3 valors

- *obtenirRGB*: rep un apuntador a una estructura `imatgeBLOCS`, el bloc i la posició del coeficient que volem transformar (coeficients de `imatgeBLOCS` en format YCbCr) Utilitzant els valors de la matriu `matriuRGB`, obté la transformació als valor originals RGB per a guardar-los en una estructura `imatgeRAW`. Retorna un apuntador a una matriu `RGB` on es guarden els 3 valors

- *dct.c*: rutines per al càlcul de la DCT (Transformada Discreta del Cosinus) Transformació per al seu ús amb doubles

- *descriptorsestadistics.c*: rep un punter a una estructura `imatgeBLOCS` on es`DCT= CERT`. Per a cada component, calcula la mitjana (m), la variància (s) i, si `r!= NULL`, el factor de forma (r) de tots els coeficients que ocupen la posició i dels arrays de 64 coeficients DCT. Retorna un punter a una estructura (`descEstadistic`) els camps de la qual són un enter que indica el nombre de components (c) i 3 arrays de

punters a tants arrays de floats com nombre de blocs hi hagi, els quals contenen els valors estadístics calculats.

- *desprocessarimatge.c*: rep un punter a `imatgeBLOCS` i retorna un punter a `imatgeRAW`. Si és una imatge monocroma s'obté el valor de color del pixel. Si no, es fa la transformació inversa de color de YCbCr a RGB Si en la transformació RGB->YCbCr es va utilitzar una matriu de transformació concreta ara cal passar l'adreça de la matriu de transformació associada, si no es fa, s'utilitza la matriu per defecte definida a *estructures.h*

- *desquantificablocs.c*: rep un punter a `imatgeBLOCS` i a un array de matrius de quantificació, una per a cada component. Si no se li passa cap punter a l'array de matrius, per defecte utilitza l'array de matrius definit a estructures.h Retorna un punter a una nova estructura `imatgeRAW` que conté el valor dels coeficients ja desquantificats

- *imatgetransforma.c*: rep un punter a `imatgeBLOCS` i transforma bloc a bloc cada un dels 64 coeficients de cada bloc. L'estructura `imatgeBLOCS` té un camp que diu si els coeficients ja han estat transformats (`esDCT= CERT`) o si encara representen els valors YcbCr Amb el paràmetre (`esDCT`) es pot triar el sentit (fer la transformada DCT o la DCT inversa) Retorna `imatgeBlocs` amb els valors dels coeficients modificats. Si `esDCT= CERT` són valors DCT. Si `esDCT= FALS` són valors YCbCr

- *llegirimatge.c*: llegeix un stream de bytes que representa un arxiu en format BMP sense compressió i l'emmagatzema en forma de taula a memòria. El pixel de dalt i de l'esquerra és el 1r element de la taula i es continua d'esquerra a dreta i de dalt a baix. Retorna un punter a una estructura de tipus `imatgeRAW` que conté les mides de la imatge (H files, W columnes), el nombre C de components (p. e. imatge monocroma C= 1, imatge RGB C= 3), el nombre de bytes per pixel (usualment B= 1) i un array dinàmic de C components on cada component és un apuntador a una seqüència de W*H bytes. Si no es pot llegir la imatge, retorna NULL.

- *llegirimatgeBMP*: rep un apuntador a l'stream de bytes que hi ha al disc (i que representa una imatge BMP) i un altre apuntador a la estructura `imatgeRAW` on volem guardar tant les dades de la mida de de l'arxiu com les dades de la imatge en

si. Les estructures BMP estan definides a bmpjordi.h, i una explicació detallada dels camps a la documentació del projecte, apartat 4

- *llegirimatgepnm.c*: rep un apuntador a l'stream de bytes que hi ha al disc (i que representa una imatge PNM i un altre apuntador a la estructura `imatgeRAW` on volem guardar tant les dades de la mida de de l'arxiu com les dades de la imatge en si. Les estructures PNM estan explicades a l'apartat 5

- *processarimatge.c*: rep un punter a `imatgeRAW` i retorna un punter a una estructura `imatgeBLOCS` que conté la mateixa informació que `imatgeRAW` amb poques variacions(H, W, C, B) En lloc d'un array dinàmic de components té un array dinàmic de llistes de blocs, on cada llista de blocs (una per cada component) és un array dinàmic de punters a bloc, i cada bloc és un array per emmagatzemar els 64 coeficients. Els coeficients ja no són els pixels originals sinó que s'ha fet la conversió de color de RGB a YCbCr. Si es vol utilitzar una matriu de transformació concreta cal passar l'adreça en invocar la funció `obtenirYCbCr`, si no es fa, s'utilitza la matriu per defecte definida a estructures.h

- *psnr2q.c*: algoritme per al càlcul d'un array de matrius de quantificació en funció d'un PSNR donat (una matriu per a cada component) Rep com a paràmetre un apuntador a una estructura desEstadistics (que guarda els descriptors estadístics de la imatge) i un float que representa el valor de PSNR que es vol aconseguir. Torna un apuntador a un array de matrius de coeficients de quantificació (conté els valors de quantificació pels quals es multiplicarà cada coeficient DCT abans de ser comprimit amb el codi de compressió corresponent) Conté les funcions necessàries per als càlculs parcials segons l'explicació del punt 7

- *taulaF.c*: rep un disposa de les funcions necessàries per a trobar la inversa de *F(t)= t/sinh(t)* També té funcions auxiliars per a crear un conjunt de 1000 valors de *F(t)= t/ sinh(t)* i una funció recursiva de cerca dicotòmica

- *testjpegq.c*: mòdul per provar la resta de mòduls de la llibreria. Crida, un per un, tots els mòduls. La crida inicial és: `tfc nomarxiu.bmp valorpsnr`

Els arxius de capçalera contenen aquelles funcions i/o estructures que poden necessitar altres mòduls per a implementar la seva funcionalitat, a excepció de estructures.h que defineix les estructures d'ús comú del projecte. S'han definit els següents arxius de capçaleres:

- `blocsquantifica.h`
- `bmpheader.h`
- `conversiocolors.h`
- `dct.h`
- `descriptorsestadistics.h`
- `desprocessarimatge.h`
- `desquantificablocs.h`
- `estructures.h`
- `imatgetransforma.h`
- `jpegq.h`
- `pbm.h`
- `pbmplus.h`
- `pgm.h`
- `pnm.h`
- `pnmheader.h`
- `ppm.h`
- `processarimatge.h`
- `psnr2q.h`
- `taulaf.h`
- `zigzag.h`

## 10.3 Com es relacionen les rutines

A continuació es descriuen les interfícies de les rutines principals en el processament de la imatge.

- `llegirimatge.c` llegeix una imatge en format BMP, o PNM, utilitzant les rutines de `llegirimatgeBMP.c`, o les de `llegirimatgePNM.c` i les guarda a la memòria

en una estructura imatgeRAW, que conté les dades de la imatge en el model de color RGB.

- `processarimatge.c` rep l'estructura `imatgeRAW` i, opcionalemnt, un apuntador a les matrius de conversió de color que es volen usar, i la transforma en una estructura `imatgeBLOCS`. Utilitza la rutina `conversiocolors.c` per a transformar les dades de la imatge al model YcbCR.

- `imatgetransforma.c` rep una estructura imatgeBMP i un indicador, `fesDCT`, de si s'ha de fer o no la transformada DCT. Obté la DCT dels coeficients de tots els components de color de l'estructura `imatgeBLOCS` i la guarda en una nova estructura `imatgeBLOCS`, en el cas en què encara no s'hagin transformat. Si ja ho han estat llavors fa la transformació inversa de la DCT. Pels càlculs fa servir la rutina `dct.c`

- `blocquantifica.c` rep una estructura `imatgeBLOCS` amb els coeficients transformats i una apuntador a un array de matrius de quantificació que ha calculat la rutina `psnr2q.`

- `desquantificablocs.c` fa la funció inversa de `blocquantifica.c`, és a dir, a partir d'una estructura `imatgeBLOCS` i un apuntador al mateix array de matrius de quantificació utilizat a `blocquantifica.c`, obté els valors dels coeficients abans de la quantificació. Només recuperarà una aproximació degut al procés previ de quantificació.

- `desprocessarimatge.c` obté, a partir de l'estructura `imatgeBLOCS` i una matriu de colors (la complementària a la utilitzada a `processarimatge.c`), una estructura `imatgeRAW` que és la que posteriorment es visualitzarà i permetrà comprovar el grau de qualitat d'imatge obtingut.

- `descriptorsestadistics.c` rep un apuntador a una estructura `imatgeBLOCS` i calcula les mitjanes, variàncies i factors de forma dels seus coeficients .

- `psnr2q.c` rep un apuntador a una estructura `descEstadistics` i calcula els coeficients d'un array de matrius de quantificació (una per cada component de color)

en funció del PSNR desitjat i del factor de resposta visual humana considerat (HVS) Té les fucnions necessàries per a calcular l'MSE. Es comunica amb `taulaF.c` i `zigzag.c` per a poder efectuar operacions necessàries per al càlcul de les matrius.

- `dct.c` rep un apuntador a una estructura `imatgeBLOCS i` calcula la Transformada Discreta del Cosinus (DCT) dels coeficients i també la transformada inversa ($DCT^{-1}$)

- `conversiocolors.c` rep un apuntador a la matriu de color que cal utilitzar i fa les conversions de color del model RGB a YcbCr i viceversa.

Les relacions entre les rutines es poden veure d'una manera gràfica a la figura 8.

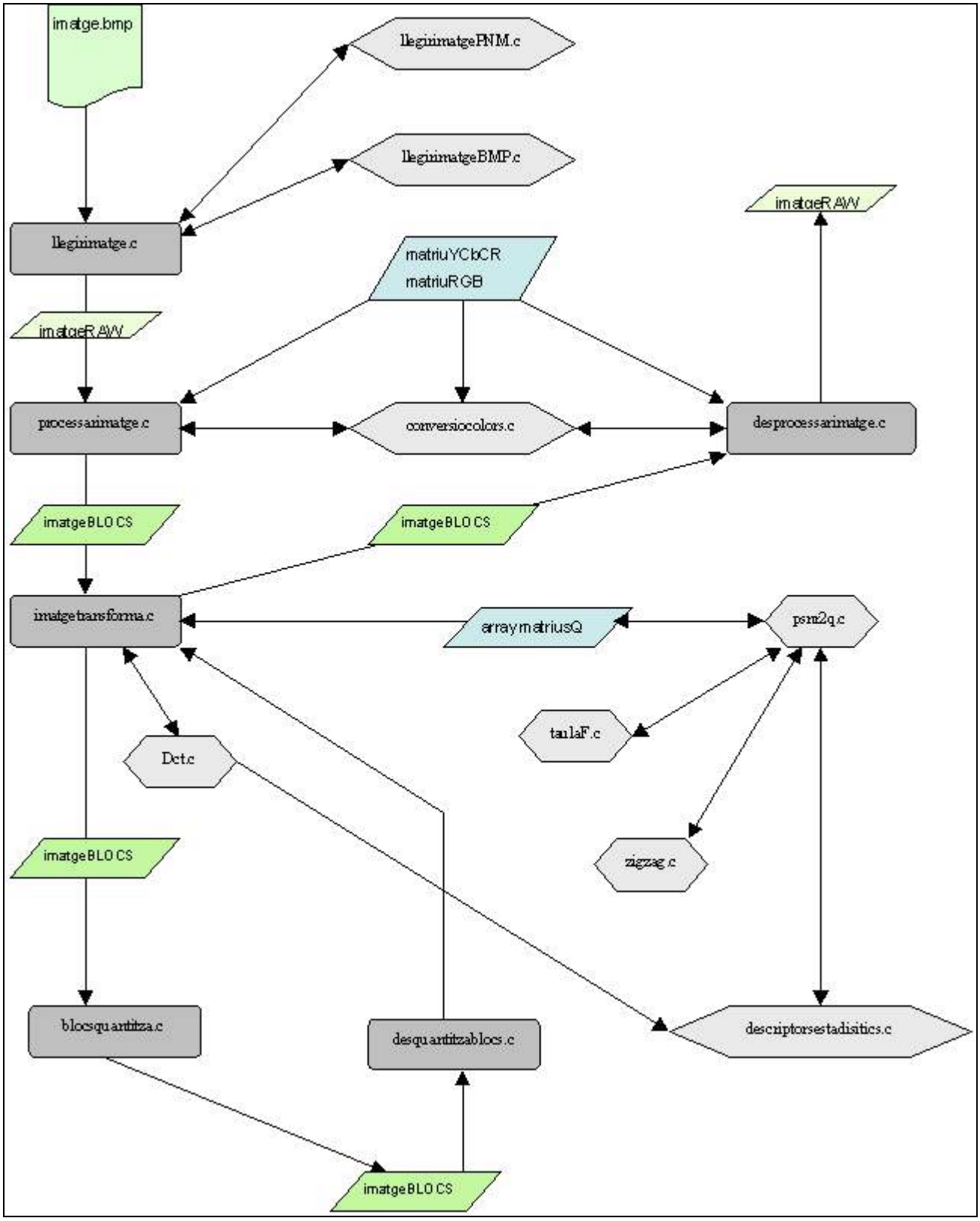



**Figura 8. Relacions entre les rutines.**

# 11 Conclusions

El conjunt de rutines implementades pretenen solucionar la problemàtica del control de qualitat d'imatges JPEG sense utilitzar el mètode d'assaig i error.

El tipus de disseny de crides entre rutines intenta obtenir un sistema fàcilment ampliable, tant en el tractament de diferents tipus de formats d'imatge com en l'ampliació dels paràmetres de control.

Sense la pretensió de ser la solució definitiva, constitueixen la base sobre la qual es pot recolzar un sistema de control de qualitat que permeti experimentar amb diverses variables abans de guardar la imatge definitivament en format JPEG.

## 11.1 Objectius assolits

S'han implementat totes les funcionalitats necessàries descrites a l'apartat 2.1, amb excepció de la interfície d'usuari, ja que el seu disseny i implementació comporta una càrrega de treball excessiva pels objectius d'aquest treball.

Les rutines estan preparades per, amb lleugers canvis, ser usades per una interfície d'usuari des de la qual es podrien modificar tots els paràmetres que afecten a la qualitat de les imatges generades.

## 11.2 Possibles ampliacions

- És necessari el disseny d'una interfície d'usuari, a poder ser en mode GUI, on es pugui modificar els diferents paràmetres i es visualitzin simultàniament les dues imatges: la original i la resultant de les modificacions.

- Lectura de qualsevol tipus d'arxiu gràfic conegut i a totes els modes dels arxius BMP. De moment hi ha suport parcial per a les imatges *PortableaNythingMap* (PNM), que engloba les *PortableBitMap* (PBM) monocromes, 1bpp; *PortableGreyMap* (PGM), d'escala de grisos: 8bpp, i *PortablePixMap* (PPM) de colors 24bpp. El codi està preparat per a ser ampliat (hi ha les funcions però estan buides).

- Afegir funcions de transformació de més models de color (en l'actualitat es suporten RGB i YcbCr).

- Gravació dels resultats obtinguts en disc (actualment només hi ha funcions per a un volcat parcial).

# 12 Bibliografia

J. Miano, *Compressed Image File Formats: JPEG, PGM, GIF, XBM, BMP.* ACM Press, 1999.

Julià Minguillón and Jaume Pujol, *"JPEG standard uniform quantization error modeling with applications to sequential and progressive operation modes"* Journal of Electronic Imaging, april 2000, vol. 11.

Gregory K. Wallace. *"The JPEG Still Picture Compression Standard"* Communications of the ACM, abril 1991. http://utenti.tripod.it/debolivo/jpeg/wallace.zip

Independent JPEG Group. *JPEG al W3C*: http://www.w3.org/Graphics/JPEG/jfif3. pdf

Independent JPEG Group. *JPEG al W3C*: http://www.faqs.org/faqs/jpeg-faq/

Deborah Olivo. JPEG: *"Metodo per la compressione di immagini"*, maig 1988, http://utenti.tripod.it/debolivo/jpeg/rel.html

Joint Photographic Experts Group. http://www.jpeg.org/public/jpeghomepage.htm

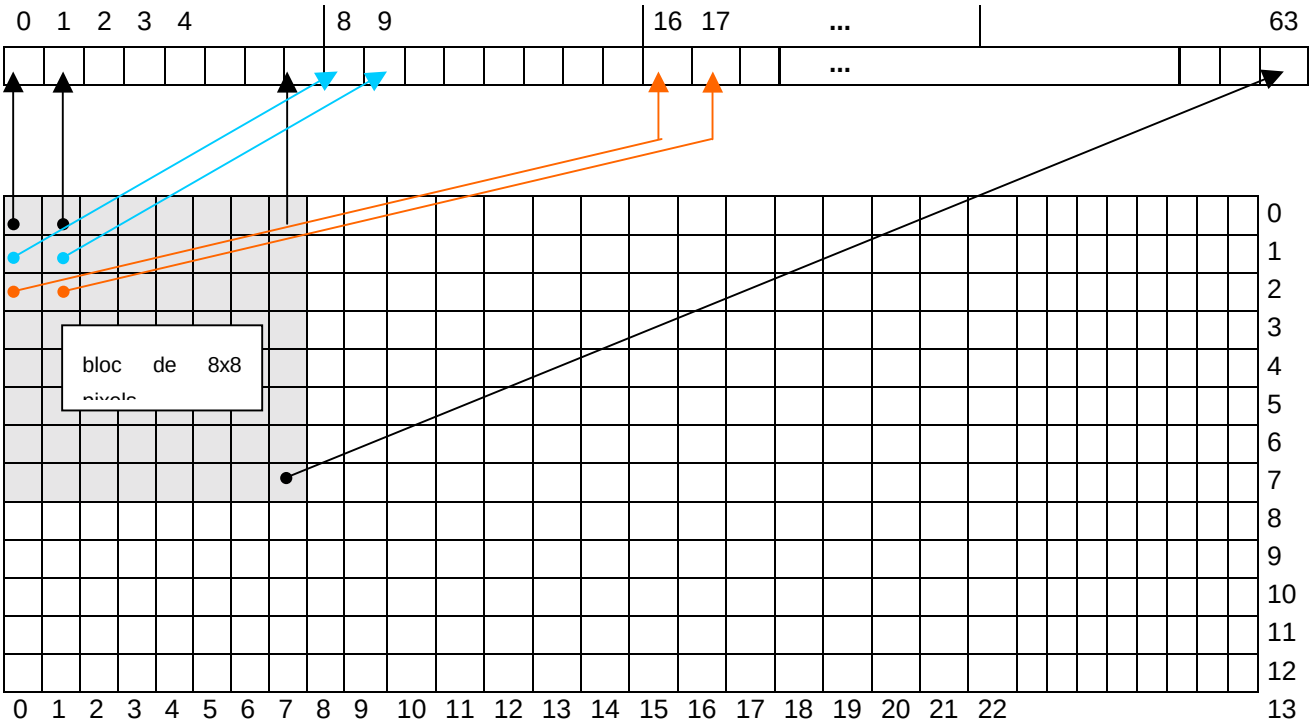
Wayne O.cochran. *"Routines for performing Discrete Cosine Transforms on 8x8x8 blocks"* Revision: 1. 1 Date: 1996/09/25 18:04:05 http://www. eecs. wsu. edu/~wcochran

Nancy J. McCracken . *"Image Format Basics"*, Setembre 1996, http://www.npac.syr.edu/users/gcf/cps606fall96/cps606image/titleabs.html

# Annexos

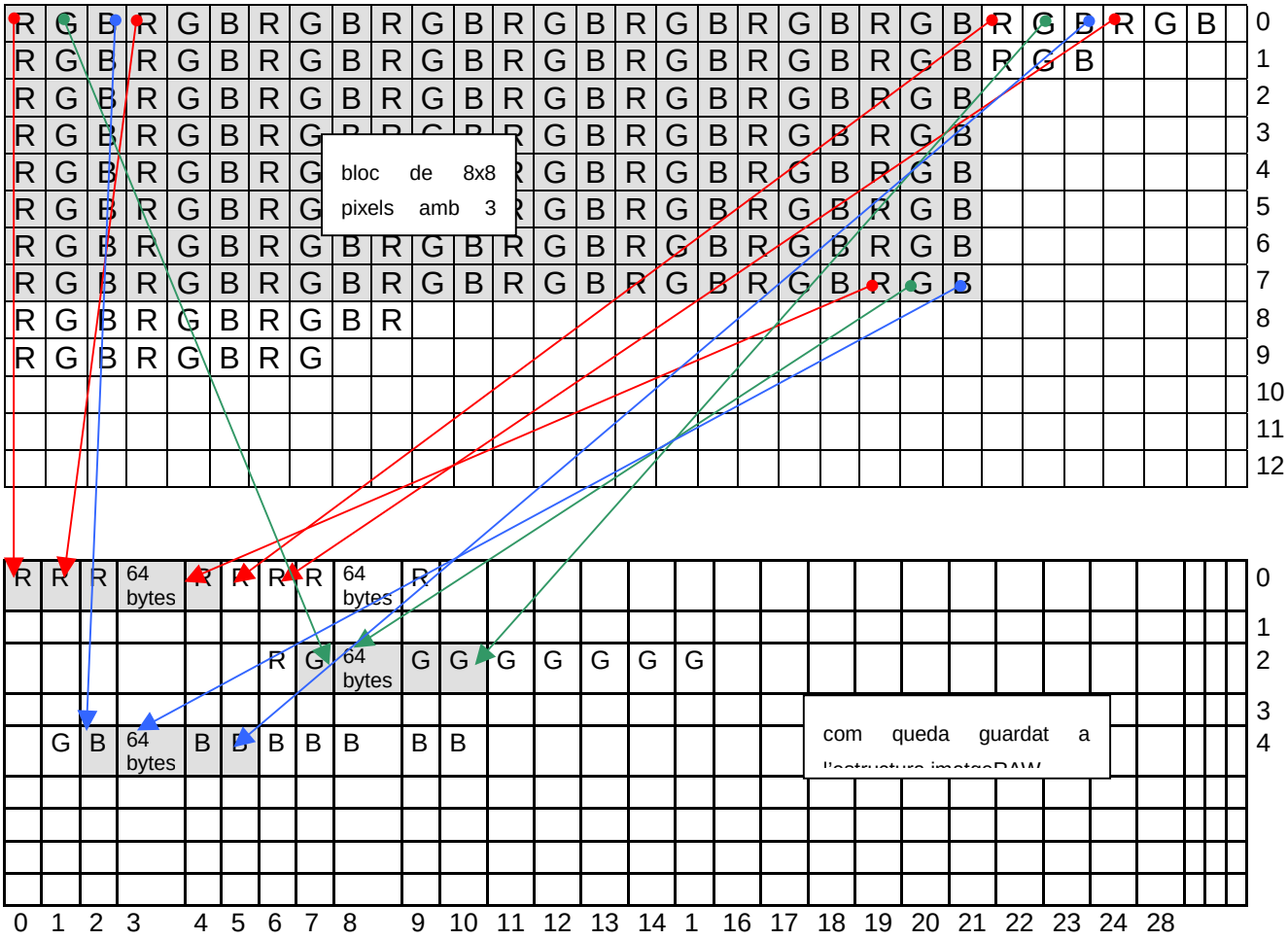
## Annex 1 Relacions entre les estructures BMP, imatgeRAW i imatgeBLOCS

### Annex 1.1 Mapejat de BMP de 8 bit (imatge monocroma) a imatgeRAW



## Annex 1.2 Mapejat de BMP de 24 bits (3 bytes per color) a imatgeRAW

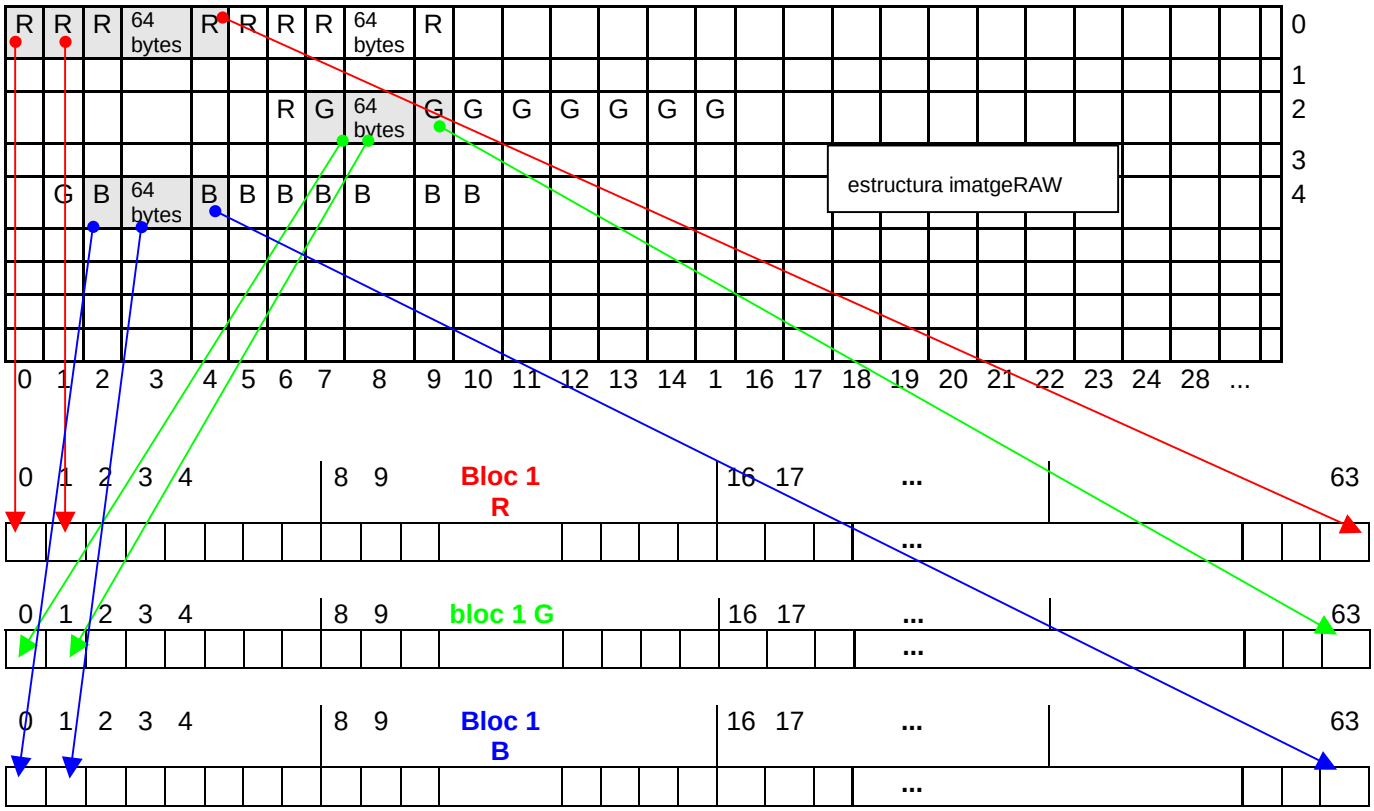
Imatge BMP




Els bytes ombrejats en gris són els 64 que corresponen al 1r bloc de cada una de les components (R, G, B)

En el cas d'una imatge amb tres components, a la imatge BMP cada pixel és representat per 3 bytes, un darrera l'altre. Després de llegir la imatge l'emmagatzemem en format RAW i queden els bytes de cada color agrupats. Com que hem de segmentar la imatge en blocs de 8x8 pixels el que farem és que, en llegir la imatge BMP i guardar-la a l'estructura imatgeRAW, els 64 pixels quedin guardats en ordre. És a dir els 64 primers bytes correspondrien a als bytes de la imatge BMP que informen de la component vermella (R) i que ocupen des de la fila 0 a la 7 i de la columna 0 a la 7. Els coeficients de cada color els guardarem en el seu bloc corresponent:

## Annex 1.3 Mapejat de imatgeRAW a imatgeBLOCS




Els bytes ombrejats en gris són els 64 que corresponen al 1r bloc de cada una de les components (R, G, B)

# Annex 2 Camps de l'estructura d'un arxiu BMP

Les següents definicions d'estructures es poden trobar a l'arxiu windows.h que s'inclou amb les llibreries del programa Visual C++.

BITMAPFILEHEADER

| inici | mida | nom | valor típic | funció |
|---|---|---|---|---|
| 1 | 2 | bfType | 19778 | ha de valer sempre 'BM' per a indicar que és un arxiu BMP |
| 3 | 4 | bfMida | ?? | especifica la mida del'arxiu en bytes. |
| 7 | 2 | bfReserved1 | 0 | ha d'estar sempre a zero. |
| 9 | 2 | bfReserved2 | 0 | ha d'estar sempre a zero. |
| 11 | 4 | bfDefBits | 1078 | especifica l'offset del bitmap de dades des del principi de l'arxiu |

BITMAPINFOHEADER

| | | | | |
|---|---|---|---|---|
| 15 | 4 | biMida | 40 | especifica la mida de la estructura BITMAPINFOHEADER en bytes. |
| 19 | 4 | biWidth | 100 | especifica l'amplada de la image en pixels. |
| 23 | 4 | biHeight | 100 | especifica l'alçada de la image en pixels. |
| 27 | 2 | biPlanes | 1 | especifica el nombre de plans del dispositiu. Ha d'estar sempre a zero |
| 29 | 2 | biBitCount | 8 | especifica el nombre de bits per pixel. |
| 31 | 4 | biCompression | 0 | especifica el tipus de compressió, normalment a zero (sense compressió) |
| 35 | 4 | biMidaImage | 0 | Especifica la mida de les dades de la imatge en bytes. Si no hi ha compressió pot estar a zero |
| 39 | 4 | biXPelsPerMeter | 0 | especifica la resolució horitzontal del dispositiu en pixels per metre, normalment a zero |
| 43 | 4 | biYPelsPerMeter | 0 | especifica la resolució vertical del dispositiu en pixels per metre, normalment a zero |
| 47 | 4 | biClrUsed | 0 | especificael nombre de colors utilitzats en el bitmap, es posa a zero si el nombre de colors es calcula a partir del camp biBitCount |
| 51 | 4 | biClrImportant | 0 | especifica el nombrer de colors que són importants pel bitmap, si està a zero tots els colors són importants |

Estructura d'una taula RGBQUAD senzilla:

| enici | mida | nom | valor típic | funció |
|---|---|---|---|---|
| 1 | 1 | rgbBlue | - | especifica la component blava |
| 2 | 1 | rgbGreen | - | especifica la component verda |
| 3 | 1 | rgbRed | - | especifica la component vermella |
| 4 | 1 | rgbReserved | - | ha d'estar sempre a zero. |

La resta de dades es troben a l'estructura BYTE i guarden la informació de color de cada pixel.